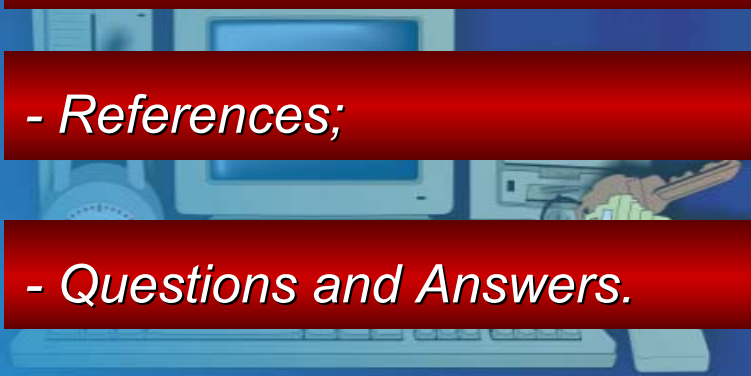
(Pseudo)random generators (RG and PRG)

Emil SIMION
e-mail: esimion@fmi.unibuc.ro

Agenda

- Cryptographic need for PRG and RG;
- Validation criteria (statistical tests and algorithmic tests);
- Implementation aspects of PRG and design for RG;
- Applications;
- References;
- Questions and Answers.



Criptological need for PRG and RG

- Difference between PRG and RG.

- Symmetric (secret) key and asymmetric (public and private) key generation;

- Message key, session key and CSP;

- Stream ciphers.

Differences between PRG and RG

- Random number generators are hardware devices, based on nondeterministic physical process which products random sequences;

- Pseudorandom generators are software/firmware implementation of a deterministic algorithm initialized with a value called ,,seed", producing a pseudorandom string.

Symmetric and asymmetric key generation

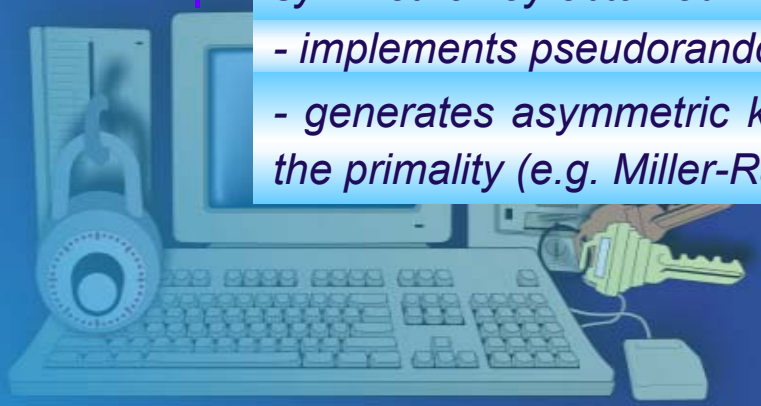
- both PRG and RG are informational - statistically tested;
- symmetric (secret) and asymmetric key generation is based on a PRG or a RG;
- for example the procedure for generation RSA asymmetric keys is the following:

1. generate (pseudo)random ,,big" odd number p (private);
2. test if p is prime number; if p is not a prime number return to STEP 1;
3. similarly generate prime number q (private);
4. N=pq is the encipher modulus (public);
5. compute Phi(N)=(p-1)(q-1);
6. enciphering modulus is number e relative prime with Phi(N);
7. deciphering modulus is d multiplicative inverse of number e modulo Phi(N).

Generation of symmetric and asymmetric key generation: Example

- Cryptographic protocol OpenSSL (Secure Socket Layer):

- OpenSSL is a multiplatform open source cryptographic package, based on a mixed structure C/C++ and Perl, which implements network protocols SSL and TLS(Transport Layer Security);
- SSL is a protocol designed by NETSCAPE which allows secure transmission of the documents over internet. Communications is session based, each session having a symmetric key obtained via a key agreement protocol.
- implements pseudorandom generators like: BBS, etc;
- generates asymmetric keys, for example RSA keys, using techniques for testing the primality (e.g. Miller-Rabin test).



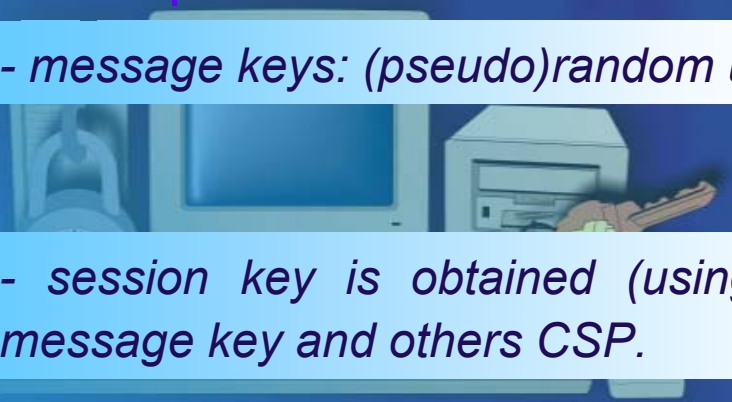
Message key, session key and others CSP

- generation of CSP (Critical Security Parameters):

  - software using PRG;

  - hardware using a device that is a RG;

- message keys: (pseudo)random using (P)RG;



- session key is obtained (using a deterministic algorithm) from secret key, message key and others CSP.

Stream Ciphers

- stream ciphering is:
  - mod 2 summation of the binary message with the output of a (P)RG;

- encryption/decryption key:
  - output of RG (in this case we deal with one time pad (OTP) system);
  - output of PRG (in this case the key is the initialization of the PRG);

- advantage & disadvantage of using (P)RG:
  - key length;
  - key life time;
  - cryptographic security: OTP is unconditionally secure;
  - etc.

Validation criteria (statistical tests and algorithmic criteria)

- statistical randomness tests (applicable PRG (stream ciphers) , RG and block ciphers);

- algorithmic criteria (applicable only PRG (stream ciphers) and block ciphers).

Statistical tests: Basic

- basic of statistical testing:
  - decision rule based on a mathematical formula which decides between two or more statistical hypothesis;
  - two kinds of statistical errors:
    - alpha = probability to reject a true hypothesis;
    - beta = probability to accept a false hypothesis;

- classification of statistical tests:
  - Neyman-Pearson which defines the most powerful test, for alpha fixed is minimized the value of beta;
  - based on confidence intervals: alpha is fixed, we build using test function a confidence region and after that we estimate the value of second order risk (beta);
  - sequential: we have three possibilities accept, reject or make a new sample;

Statistical tests: interpretation

The concept of statistical testing can be found in every statistical book. Statistical testing in cryptography can be found in *Maurer* [8]. Here we have two statistical hypothesis regarding the binary sequence which is under test:

H_0 : the sequence $\mathbf{x}$ is produced by a binary memory less source: $Pr(X = 1) = p_0$ and $Pr(X = 0) = 1 - p_0$, (in this case we say that the sequence does not present any predictable component)

and the alternative:

H_1 : the sequence $\mathbf{x}$ is produced by a binary memory less source: $Pr(X = 1) = p_1$ and $Pr(X = 0) = 1 - p_1$ with $p_0 \neq p_1$,(in this case we say that the sequence present a predictable component regarding the probability p).

In statistical testing are two kinds of errors: the first order error denoted by α (also called level of significance, it is the probability of occurrence of a false positive result) and second order of error denoted by β (is the probability of the occurrence of a false negative result). This errors have the following interpretation:

$$\alpha = \Pr(\text{reject } H_0 | H_0 \text{ is true}) = 1 - \Pr(\text{accept } H_0 | H_0 \text{ is true})$$

and

$$\beta = \Pr(\text{accept } H_0 | H_0 \text{ is false}) = 1 - \Pr(\text{reject } H_0 | H_0 \text{ is false}).$$

This two errors can't be minimized simultaneous (Neymann-Pearson tests minimize the value of β for a given α). The testing procedure we present here is the following: for a fixed value of α we find a confidence region for the test statistic and check if the test statistic value is in the confidence region. The confidence region is computed using the quantiles of order $\frac{\alpha}{2}$ and $1 - \frac{\alpha}{2}$. (For example the quantile u_α of order α is defined by $Pr(X < u_\alpha) = \alpha$.) Let us denote f_{test} the value of test function. Another equivalent method is to compare the $P - value = Pr(X < f_{test})$ with α and decide the randomness if $P - value \geq \alpha$.

Statistical test: decision rule

Example of decision rule, based on P-value:

P-value is the smallest significance value (first order risk) for which we reject H_0 (based on the value of statistics $f(X_1,\ldots,X_n)$ thus:

if $P \leq \alpha$ reject H_0 at significance level α

and

if $\alpha > P$ accept H_0 at significance level α.

Statistical tests: Examples

- examples of statistical tests:
  - frequency test;
  - serial test;
  - runs test;
  - autocorrelation test;
  - poker test;
  - entropy (Shannon, Renyi etc.) test;
  - Berlekamp-Massey (liniarity) test;
  - compression tests: Maurer, Lempel-Ziv;
  - etc.;

- statistical randomness standards:
  - SP 800-22 NIST Statistical Test Suite;
  - Diehard Test Suite.

Algorithmic criteria - 1

- useful only for PRG and block ciphers;
- assume to build a function with m bits input and n bits output.

- examples of tested criteria:
  - balance: every n-bits output is produced by the same number of m-bits input;
  - correlation immune: correlated inputs produce uncorrelated outputs;
  - strict avalanche criteria: changing one bit at input we have 50% (in mean value) changing at output;
  - nonlinearity: unction is nonlinear;
  - nondegeneracy: function depends effectively of all inputs;
  - minimum period P;
  - linear complexity LC.

Algorithmic criteria - 2

- testing algorithmic criteria are based on WALSH-HADAMARD transform.

- WALSH-HADAMARD transform is defined in the following way:

Let us consider the binary function $f : \mathbf{Z}_2^n \to \mathbf{Z}_2$ written in algebraic normal form. We define $\hat{f}$ transform by the formula

$$\hat{f}(\mathrm{x}) = 1 - 2f(\mathrm{x}) = (-1)^{f(x)},$$

thus $\hat{f}: \mathbf{Z}_2^n \to \{-1, 1\}$.

Definition *For a function* $\hat{f}: \mathbf{Z}_2^n \to \{-1, 1\}$ *we define the Walsh-Hadamard transform* $\hat{F}$ *by the formula*

$$\hat{F}(\omega) = \sum_{\mathrm{x} \in \mathbf{Z}_2^n} \hat{f}(\mathrm{x})(-1)^{\omega.\mathrm{x}} \text{ for every } \omega$$

where $\omega.\mathrm{x}$ *is the scalar product of* ω *and* x. *Note that the Walsh Hadamard transform can be defined for any real function.*

Implementing (P)RG

- implementation of PRG:

- using combination of LFSR/NLFSR (linear/nonlinear feedback shift register);

*- evaluate the period **P**;*

*- evaluate linear complexity **LC**;*

- statistical testing;

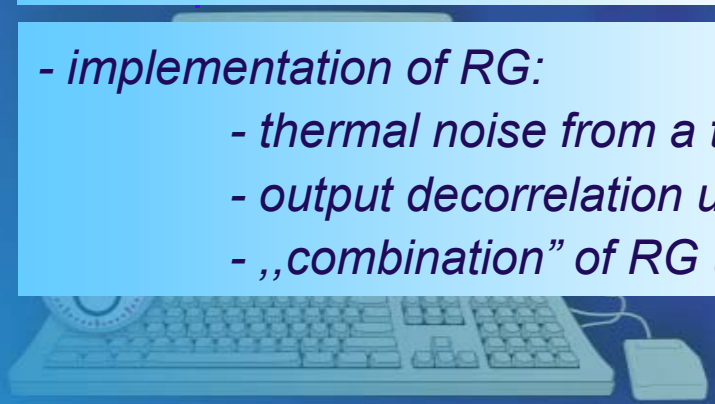
- informational testing (algorithmically testing).

- implementation of RG:

- thermal noise from a transistor;

- output decorrelation using von Newman technique;

- ,,combination" of RG using some algebraic functions;



Some possible applications

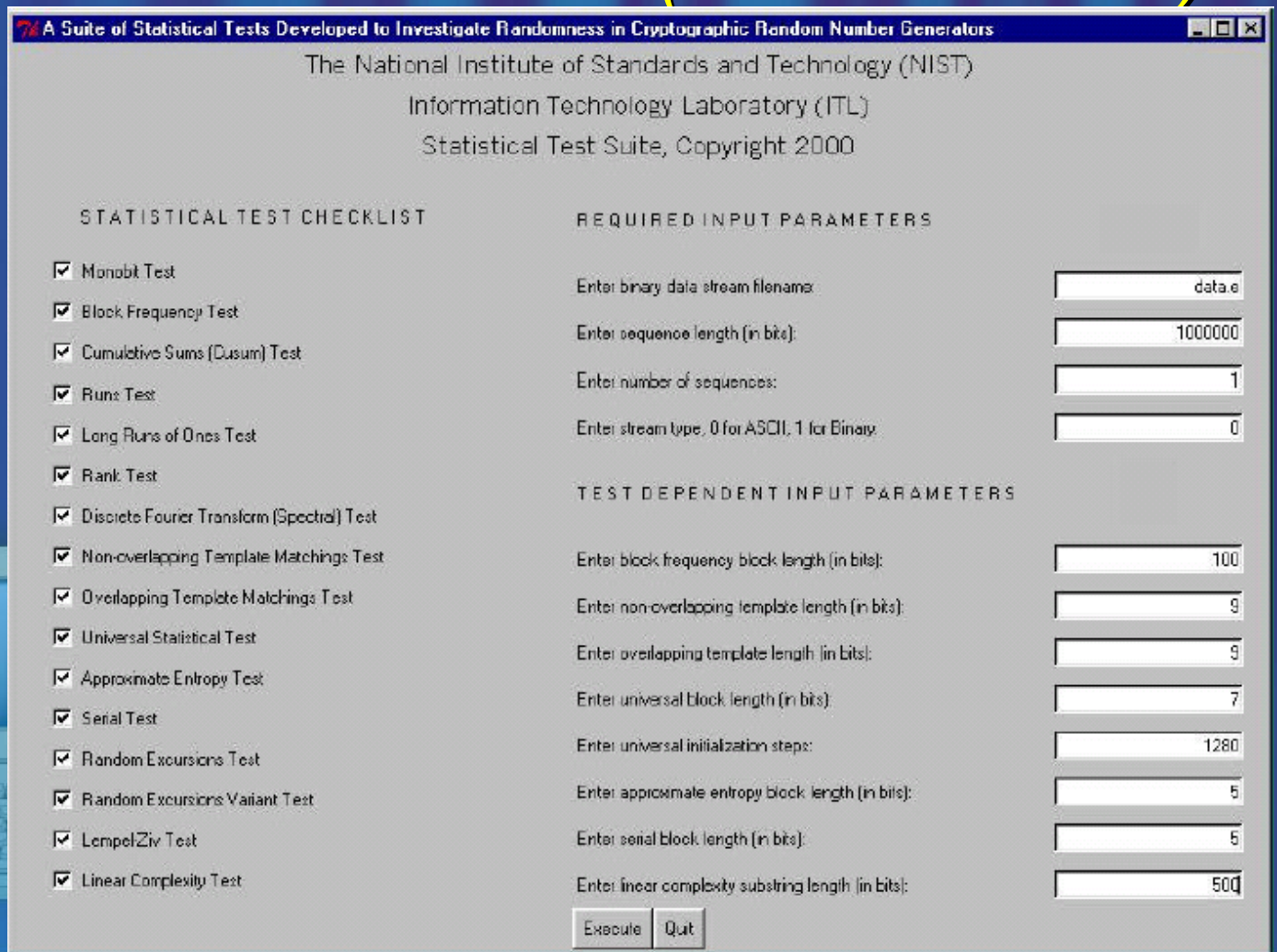
1. Statistical testing package NIST 800-22;
2. Frequency test:
   - test function;
   - implementation;
   - interpretation of the results.
3. Walsh-Hadamard transform. Definition & Application;
4. Evaluation of pseudorandom:
   - linear feedback shift registers;
   - Geffe generator;
   - Beth&Piper generator.
5. Evaluation of stream ciphers:
   - A5 GSM standard;
   - caracteristici statistico-algoritmice;
6. Evaluation of block ciphers:
   - AES (Advanced Encryption Standard);
   - statistical tests applied to block ciphers;
   - evaluation criteria of block ciphers.

NIST - STS SP 800-22

- Statistical testing package designed by NIST (National Institute for Standards and Technologies) published in NIST SP 800-22;
- Is composed from 16 statistical tests based on confidence intervals;
- In NIST SP 800-22 is presented the pseudocode and the description of each statistical test. Also there are some testing strategies and interpretation schemas;
- The test detects a general class of defects of (pseudo)random generators;
- NIST give the source code and a software which has a GUI (in fact there are some versions available);
- Rejection rate was setup at 1%.

NIST -STS

SP 800-22 GUI (former version)



Frequency test: Description & Purpose

Test description: The focus of the test is the proportion of zeroes and ones for the entire sequence. The purpose of this test is to determine whether the number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence. The test assesses the closeness of the fraction of ones to $p_0 = 0.5$, that is, the number of ones and zeroes in a sequence should be about the same. All subsequent tests depend on the passing of this test (reference test or basic test) ; there is no evidence to indicate that the tested sequence is non-random. The test can be performed at different rejection rates and different values of p_0 (that is non uniform testing purposes).

Test purpose: Frequency test indicates if there is a significativelly deviation of the probability (real) of the symbols of 1 from the theoretical probability (ideal).

Test statistics and reference distribution: Test statistics is the normalised sum of the bits from the sequence. The reference distribution for the test statistic is half normal;

Mathematically speaking the test statistics is binomial but when the number of observations tents to infinit this distribution is well approximed by normal distribution.

In next slides we present the pseudocode and an example of application.

Frequency test: algorithm

Input: Binary sequence $s^N = s_1, \ldots, s_N$.

Probability p_0 of occurrences of symbol 1 (if $p_0 = 0,5$, then we have a binary symmetric source).

Let us denote by $q_0 = 1 - p_0 = \Pr(S = 0)$ the probability of occurrences of symbol 0.

Output: The decision of acceptation or rejection of randomness, that is sequence s^N is the output of a symmetric binary source with $\Pr(S = 1) = p_0$, the alternative hypothesis being $\Pr(S = 1) = p_1 \neq p_0$.

STEP 0. Read the sequence s^N and the rejection rate α.

STEP 1. Compute the test function:

$$f_{T_F}(s^N) = \frac{1}{\sqrt{N * p_0 * q_0}}(\sum_{i=1}^{N} s_i - N * p_0).$$

STEP 2. If $f_{T_F}(s^N) \in [u_{\frac{\alpha}{2}}; u_{1-\frac{\alpha}{2}}]$ ($u_{\frac{\alpha}{2}}$ and $u_{1-\frac{\alpha}{2}}$ are the quantiles of normal repartition of order $\alpha/2$ respectively $1 - \alpha/2$) we accept the hyphothesis of randomness, else we reject the hyphothesis of randomness.

STEP 3. Compute the risk of order 2 (probability of acceptance a false hyphothesis):

$$\beta = 1 + \Phi((u_{\frac{\alpha}{2}} - \frac{N(p_1 - p_0)}{\sqrt{N p_0 q_0}})\sqrt{\frac{p_0 q_0}{p_1 q_1}}) - \Phi((u_{1-\frac{\alpha}{2}} - \frac{N(p_1 - p_0)}{\sqrt{N p_0 q_0}})\sqrt{\frac{p_0 q_0}{p_1 q_1}}),$$

where $\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{x^2}{2}} dx$ is *Laplace function.*

Frequency test: numerical example

Input data: p_0=0,5, rejection rate of the test alpha=0,05, binary sequence of length N=4096, $\mathbf{s}=s_0,\ldots,s_{4095}$.

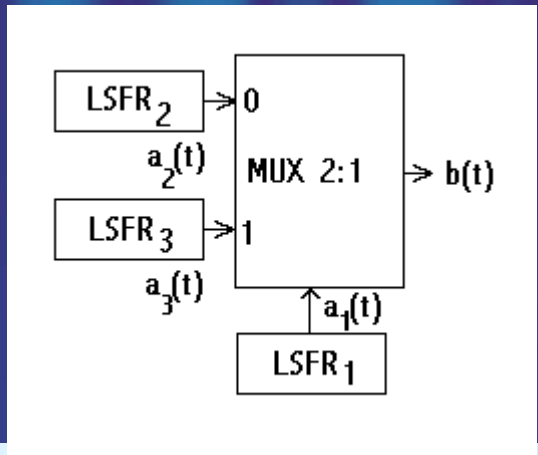
Output data: decision regarding the randomness of sequence **s**;

Critical region: In determined by the quantile of the normal distribution $u_{0,025}$=2,57, minimum number of symbols of 1 is l_{inf}=1965 and maximum number of symbols of 1 is l_{sup}= 2130;

Test statistics: Computes the number of 1 from the sequence **s**: 2100;

Decision regarding the randomness: if $l_{inf} < \mathbf{s} < l_{sup}$ accept that the sequence is random else reject the hypothesis of randomness: In this case we accept the hypothesis of randomness because the tested sequence have the numbers of 1 in acceptance region defined by l_{inf} and l_{sup} ;

Example: Geffe's generator



Characteristics:

*- is implemented using three linear feedback shift registers **LSFR** using multiplexing technique;*

- analytic function: $b(t) = a_1(t)a_3(t) + a_1'(t)a_2(t) = a_2(t) + a_1(t)(a_2(t) + a_3(t))$, where + is modulo 2 summation operator and ' is the complementation (not) operator;

*- linear complexity operator: **LC**= $n_3n_1 + (n_1 + 1)n_2$, where n_i are the degree of the three feedback primitive polynomials;*

*- minimal period: **P**=$(2^{n1}-1)(2^{n2} - 1)(2^{n3} - 1)$, if n_i are relative prime.*

Geffe's weak points

The weakness *of this generator consist from probability of coincidence (different from 0.5):*

$$p = Pr(b(t) = a_2(t)) = Pr(a_1(t) = 0) + Pr(a_1(t) = 1)Pr(a_3(t) = a_2(t)) = 0.75.$$

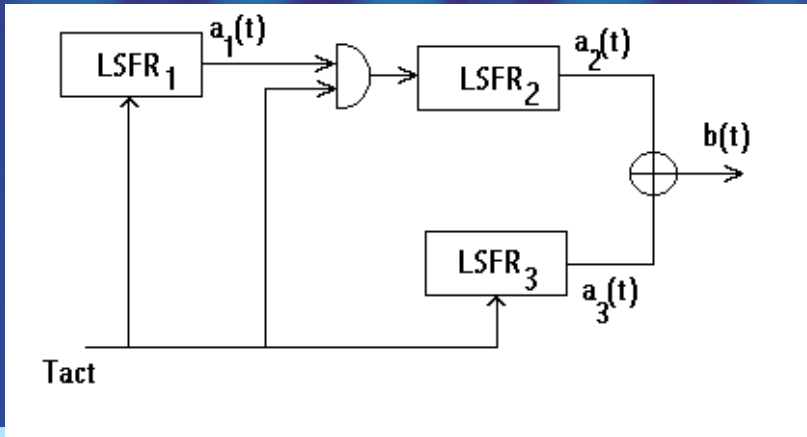
Probability of coincidence between b(t) and a_3 (t) can be estimated in a similar way;

Thus if the feedback primitive trinomials of the LFSR are known, each of them with degree smaller then n, then we can crack the Geffe's generator using linear syndrome method;

The state of the three LFSR can be revealed from a intercepted string of length N = 37n from the output;

Computation cost is about 896 operations per n bits.

Example: Beth & Piper generator



Characteristics:

- is implemented using three linear feedback shift registers ***LSFR*** *using AND and OR gates;*

- analytical function: $b(t) = a_3(t) + a_2(t)$ with the clock of $a_2(t)$ controlled by $a_1(t-1)$, where + is the summation mod 2 operator 2;

*- linear complexity: **LC** $=(2^{n1}-1)n_2 + n_3$, where n_i are the degree of the three primitive polynomials;*

*- minimum period: **P**$=(2^{n1}-1)(2^{n2}-1)(2^{n3}-1)$, if n_i are relative prime.*

Beth & Piper generator – week points

In intermittent generators information is not moving at each time t;

In this case each generator from the system has a clock which decides if the information is moving or stay on the same position in the register;

Beth- Piper has three ***LFSR****, with the output sequences* a_1, a_2 *, respectively* a_3*, where* $LFSR_1$ *and* $LFSR_3$ *have the same clock, the clock of* $LFSR_2$ *is controled by* $LFSR_1$*, such that* $LFSR_2$ *is shifted at time t if and only if* $a_1(t-1) = 1$*;*

The output sequence has a huge linear complexity but the cryptographic security is low;

Let us suppose that $a_2'(t)$ *is the output* $LFSR_2$ *in the case* $a_1 = 1$*;*

Then we have the coincidence probability:

$p = Pr(b(t) + b(t+1) = a_3(t) + a_3(t+1)) = Pr(a_1(t) = 0) + Pr(a_1(t) = 1)Pr(a_2'(t) = a_2'(t+1)) = 0.75$, *where* **+** *is modulo 2 summation.*

References

1. Menenzes A.J., et. al., Handbook of Applied Cryptography, CRC Press, 1997.
2. Schneier B., Applied Cryptography, John Wiley & Sons, Inc. 1998.
3. Simion E., Preda V., Popescu A., Criptanaliza. Rezultate si Tehnici Matematice, Ed. Universitatii Bucuresti, 2004.
4. Simion E., Cifruri Flux, Lectures notes, 2008.
5. Simion E., Analiza Statistico-Informationala, Lectures notes, 2008.
6. Stallings W., Cryptography and Network Security: Principles and Practice, Prentice Hall, Second Edition, 1999.
7. Tilborg, Henk C.A. van, Fundamentals of Cryptology, Kluwer Academic Publisher, Second Edition, 2001.

Questions and Answers